****

# Artemis Financial Vulnerability Assessment Report

Table of Contents

[Document Revision History 3](#_Toc32574607)

[Client 3](#_Toc32574608)

[Instructions 3](#_Toc32574609)

[Developer 4](#_Toc32574610)

[1. Interpreting Client Needs 4](#_Toc32574611)

[2. Areas of Security 4](#_Toc32574612)

[3. Manual Review 4](#_Toc32574613)

[4. Static Testing 4](#_Toc32574614)

[5. Mitigation Plan 4](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **1/20/2023** | **Tyler Primas** | **Fully filled out vulnerability report** |

## Client



## Developer

Tyler Primas

## Interpreting Client Needs

Artemis Financial is a consulting company that develops individualized financial plans for their customers, so making sure those financial plans are kept private is important to them. These financial plans contain information on savings, retirement, investments, and insurance, so keeping that information safe and secure should be a high priority for the company.

Artemis Financial does not specify if they do international transactions or not, so we should find out that information to know if there are specific rules and regulations based on where the transactions may take place.

There are some governmental standards to consider, like PCI-DSS for credit card management if the company is dealing with credit card payments. Another important standard to follow is Gramm-Leach-Bliley for financial information management. These standards make it so customers are aware of the practices we are following to keep their information secure from external forces.

There will always be the external threat of people trying to hack in and retrieve information. There is the potential that a hacker could steal money through hacking, which makes it very desirable for them to attempt to gain access to the system.

When modernizing and using open-source libraries, it is important consider that those libraries could be vulnerable to security issues. If there is a security issue within the code of these libraries, we may have to wait on them to update and fix the issues rather than being able to fix it on our own. This could potentially cause many security problems, so it is important to keep the libraries up to date and try to use reputable libraries that have a good history of relatively few issues. Evolving web application technologies should be taken into account when modernizing as well by developing a system that can be easily updated and modified to become more secure and well designed.

## Areas of Security

The areas of security I would focus on for the software application are APIs, input validation, and cryptography. Because we will be using a RESTful API, we need to make sure that the interactions of the API are secure. APIs are commonly used for many different applications, so they have become more susceptible to hacking attempts and should be monitored closely to make sure there are as few ways to hack the system as possible. Input validation is important because making sure the input is secure will prevent unauthorized access to the systems. It makes sure that only data that is acceptable to be inputted is accepted and prevents against various attacks like SQL injection. Cryptography is important especially for this application because it uses encryption to be able to follow governmental standards like Gramm-Leach-Bliley to keep user data secure. This way input from the user will be encrypted so that if a hacker gains access to the system, the encrypted data does not give away the user’s information.

## Manual Review

The CRUDController.java file has value= ”business\_name” set as a parameter and takes the business name given by the user. This could lead to injection and should be changed to prevent that from happening. In the customer.java file, the int account\_balance is not set to private and should be as to not give away too much information about the user’s finances. The account balance and account number should also probably have encryption on it to keep that information from getting into the hands of someone that wants to steal it. The DocData.java file looks like it has a method to read a document in it and within that method there is hard coded username and password. Putting usernames and password into the code is dangerous and an easy way for someone to gain access to the system. The myDateTime.java file is not completed and need accessor methods implemented the retrieve and set the time. Input validation may need to be implemented in this to make sure the user does not input potentially harmful inputs.

## Static Testing

Graphical user interface

Description automatically generated with low confidence

The following descriptions of each dependency as listed above, and the vulnerability documented:

The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains JCE provider and lightweight API for the Bouncy Castle Cryptography APIs for JDK 1.5 to JDK 1.7: many unsafe modes and invalid keys, cryptographic issues

Spring Boot: vulnerable to temporary directory hijacking

Logback-core module: attacker could create malicious configuration and execute dangerous code

The Apache Log4j API: improper validation of certificate

YAML 1.1 parser and emitter for Java: DoS issues and can consume large amounts of system resources

General data-binding functionality for Jackson that works on core streaming API: not properly secured

Core Tomcat implementation: refactoring issues: attacker could control contents and names on files

Hibernate’s Bean Validation (JSR-380) reference implementation: allows attackers to bypass input sanitation

Spring Web: potential for malicious input

Spring Beans: vulnerable to remote code execution, code injection

Spring Web MVC: malicious input could log entries

Spring Context: improper handling of case sensitivity

Spring Expression Language (SpEL)

The recommended solution for almost every vulnerable dependency is to update each to the newest version.

## Mitigation Plan

After looking at the dependencies that are vulnerable, updating those should be number one priority to fix many of the issues they have. Input validation should be added into the code to make sure that the input from the user is not being used in a way that could lead to injection and giving an attacker access to information they should not have. Encryption should be introduced for things like account information and passwords for extra security to protect the user and their information.